Amendments to the Claims:

The listing of Claims will replace all prior versions and listings of the Claims in the application:

Listing of Claims

1. (Currently Amended) A method of interfacing a front-end systems layer with a back-end systems layer using a self describing data structure, the method comprising:

a) receiving a request from a front-end systems layer;

b) translating the request;

c) executing custom application code to access data within a back-end systems layer based on~~as a function of~~ the translated request;

d) receiving data in response to the translated request from the custom application code; and

e) translating the data to a format defined in the request.

2. (Currently Amended) The method of claim 1, ~~wherein a) comprises~~ further comprising the initial step of generating the request in a servlet request format.

3. (Original) The method of claim 1, wherein b) comprises translating the request to extensible markup language.

4. (Original) The method of claim 1, wherein b) comprises translating the request into a document object model document to represent an input message.

5. (Original) The method of claim 1, wherein b) comprises limiting the translated request to representation as at least one of integer, long, Boolean, string and group fields.

6. (Original) The method of claim 1, wherein b) comprises generating a plurality of fields as a function of tags provided in the request.

7. (Original) The method of claim 1, wherein b) comprises selectively setting the length of a field name for each of a plurality of fields.

8. (Currently Amended) The method of claim 1, wherein cd) comprises establishing a structure for athe response in extensible mark up language.

9. (Original) The method of claim 1, wherein d) comprises limiting the data to representation as at least one of integer, long, Boolean, string and group.

10. (Currently Amended) The method of claim 1, wherein d) comprises generating a document object model document to represent an output message based on ~~as a function of~~ data received.

11. (Original) The method of claim 1, wherein d) comprises translating to one of hypertext markup language, extensible markup language and website meta language.

12. (Original) The method of claim 1, further comprising f) returning the translated data to the front-end systems layer.

13. (Currently Amended) A method of leveraging extensible markup language technology to interface a front-end systems layer with a back-end systems layer, the method comprising:

a) receiving a request initiated with a delivery technology;

b) identifying the value of a request name parameter from the request;

c) translating the request to an input message, the input message comprising a root element and a plurality of sub elements; and

d) initiating the retrieval of data based on ~~as a function of~~ the request name parameter.

14. (Original) The method of claim 13 further comprising:

e) providing data as a response;

f) creating an output message with the response, the output message comprising a root element and a plurality of sub-elements; and

g) translating the output message to a format compatible with the delivery technology.

15. (Original) The method of claim 13, wherein c) comprises setting the root element to a message name as a function of the request name parameter.

16. (Original) The method of claim 13, wherein c) comprises creating a document object model document.

17. (Original) The method of claim 13, wherein d) comprises executing custom application code corresponding to the request name parameter.

18. (Original) The method of claim 14, wherein f) comprises creating a document object model document.

19. (Original) The method of claim 14, wherein f) comprises setting the root element to a message name as a function of the request name parameter.

20. (Original) The method of claim 13, wherein the delivery technology comprises at least one of an Internet browser, a telephone, a wireline communication device, a wireless communication device and a wireless application protocol device.

21. (Currently Amended) A method of operating a business services application for retrieving data with delivery technologies, the method comprising:

a) developing custom application code in a subclass of a BusinessService class, the custom application code responsive to a request for data initiated by the delivery technologies;

b) translating the request to a first document object model document with an ApiService class;

c) selectively limiting the data structure of the first document object model document with ~~as a function of~~ a Message class and a Field class;

d) executing the custom application code to retrieve data based on ~~as a function of~~ the first document object model document;

e) reading data into a second document object model document with the ApiService class;

f) selectively limiting the data structure of the second document object model document with ~~as a function of~~ the Message class and the Field class; and

g) translating the second document object model document with the ApiService class based on ~~as a function of~~ the delivery technology.

22. (Original) The method of claim 21, wherein c) comprises setting a plurality of text nodes within the first document object model document to a unit of data identified by a tag in the request.

23. (Original) The method of claim 22, wherein c) further comprises limiting the unit of data to a predetermined datatype.

24. (Original) The method of claim 23, wherein c) further comprises limiting the predetermined datatype to a string.

25. (Original) The method of claim 21, wherein c) comprises setting an attribute node within the first document object model document to an attribute identified by a request name parameter in the request.

26. (Original) The method of claim 21, further comprising selecting, as a function of a mode debug flag, one of a short field name and a long field name for each of a plurality of fields in the first and second document object model documents.

27. (Original) The method of claim 21, wherein b) comprises representing an input message with the first document object model document.

28. (Original) The method of claim 21, wherein e) comprises representing an output message with the second document object model document.

29. (Currently Amended) The method of claim 21, wherein f) comprises setting, based on ~~as a function of~~ a datatype, a plurality of text nodes within the second document object model document to data read in to the second document object model document.

30. (Original) The method of claim 21, wherein f) comprises setting, as a function of a datatype, an attribute node within the second document object model document to an attribute read in to the second document object model document with the data.

31. (Original) The method of claim 30, wherein the attribute comprises an attribute name and an attribute value and f) further comprises limiting the attribute value to a predetermined datatype.

32. (Original) The method of claim 21, wherein g) comprises translating the second document object model document to extensible markup language text.

33. (Original) The method of claim 21, wherein g) comprises translating the second document object model document to at least one of a hypertext markup language and a website meta language as a function of at least one extensible stylesheet language stylesheet.

34. (Currently Amended) An e-commerce ~~software~~ architecture for providing a framework to interface delivery technologies with data, the e-commerce ~~software~~ architecture comprising:

a server computer operable to execute instructions to convert a request to an input message in a predetermined extensible markup language format, the input message comprising a plurality of request parameters,

the server computer operable to execute instructions to retrieve data as a function of the request parameters,

the server computer operable to execute instructions to create an output message in a predetermined extensible markup language format, the output message comprising the data retrieved, and

the server computer operable to execute instructions to convert the output message to a format indicated by the request.

35. (Currently Amended) The e-commerce ~~software~~ architecture of claim 34, wherein the request comprises a servlet request format.

36. (Currently Amended) The e-commerce ~~software~~ architecture of claim 34, wherein the predetermined extensible markup language format of the input message and the output message is predetermined as a function of instructions comprising a createField method and a setAttribute method.

37. (Currently Amended) The e-commerce ~~software~~ architecture of claim 34, wherein the instructions to convert a request to an input message comprises a createInputMessage method, a createField method and a setAttribute method.

38. (Currently Amended) The e-commerce ~~software~~ architecture of claim 34, wherein the instructions to create an output message comprises a createOutputMessage method, a createField method and a setAttribute method.

39. (Currently Amended) The e-commerce ~~software~~ architecture of claim 34, wherein the instructions to retrieve data comprises custom application code.

40. (Currently Amended) The e-commerce ~~software~~ architecture of claim 34, wherein the instructions to convert the output message comprises one of a generateXML method and a generatePresentation method.

41. (Currently Amended) A system~~business services layer~~ for leveraging extensible markup language technology to provide an interface between a back-end systems layer and a front-end systems layer, the system~~business services layer~~ comprising:

a server computer;

an ApiService class operable within the server computer to direct the translation of a request to an input message;

a document object model class operable within the server computer to represent the input message as a document object model document;

a Message class and a Field class operable within the server computer as wrapper of the document object model class to restrict manipulation of the document object model document; and

a BusinessService class operable within the server computer to direct the execution of custom application code as a function of the input message.

42. (Currently Amended) The system~~business services layer~~ of claim 41, wherein the custom application code is operable to process the input message to retrieve data, the data translatable with the document object model class, the Message class and the Field class to an output message in the form of a document object model document.

43. (Currently Amended) The system~~business services layer~~ of claim 42, wherein the ApiService class is operable to direct the conversion of the output message to a presentation format defined by the request.

44. (Currently Amended) The system~~business services layer~~ of claim 41, wherein the input message and the output message comprises a root element and a plurality of sub-elements.

45. (Currently Amended) The system~~business services layer~~ of claim 41, further comprising a Fldtypes class operable within the server computer, wherein the Fldtypes class comprises definitions of the format of datatypes for fields within the input message.

46. (Currently Amended) The system~~business services layer~~ of claim 41, wherein the document object model document comprises a plurality of field names, the field names selectable with a mode debug flag as one of a first field name and a second field name.

47. (Currently Amended) The system~~business services layer~~ of claim 46, wherein the first field name and the second field name are defined in a MESSAGEDEFINITION class operable within the server computer.

48. (Currently Amended) The system~~business services layer~~ of claim 41, wherein the document object model class comprises a Document class, a document object model Element class and a plurality of ProcessingInstruction classes, the Message class operable as a wrapper of the Document class, the document object model Element class and the Processing Instruction classes.

49. (Currently Amended) The system~~business services layer~~ of claim 41, wherein the document object model class comprises a document object model setAttribute method, Field class operable as a wrapper of the document object model setAttribute method.

50. (Currently Amended) The system~~business services layer~~ of claim 41, wherein the BusinessService class comprises a subclass of custom application code responsive to the request.